IN THE SPECIFICATION:

Please amend the paragraphs from page 6, line 24 to page 7, line 18, as shown below.

Figure 1 illustrates some examples of data sources which may be bookmarked. As described above, the bookmark service may operate in conjunction with an open activation framework allowing various types of data sources to be typed and encapsulated. Figure 1 illustrates some exemplary types of data sources which users of typical application programs running on a small footprint device may wish to bookmark.

As illustrated in Figure 1, a user may bookmark a particular piece of email 12. For example, the user may select a particular command while reading an email on an email client 10 which creates a bookmark for the email. The user may later quickly refer to the email using the bookmark. For example, the user may invoke a bookmark service which displays a list of bookmarks. Figure 1 shows other data sources which may be bookmarked, such as a web page 22 bookmarked from within a web browser 20 and an appointment entry 32 bookmarked from within a personal information manager program 30.

Bookmarks may be organized in various ways. For example, a system may have a central list 40 where all bookmarks are kept, or separate applications may have their own bookmark lists, or various combinations of these approaches may be taken. The request to create a bookmark may occur in various situations, such as invoking a bookmark service and selecting a data source to bookmark, or issuing a command from within an application which invokes a bookmark service, etc. In one embodiment, bookmarks may be imported from another system.

The bookmarks of Figure 1 may reference local or remote data sources. For example, the web page bookmark shown may reference an HTML page stored on a remote server, or the email bookmark shown may reference an email stored on another system.

Please amend the paragraphs from from page 9, line 26 to page 11, line 5, as shown below.

Figure 4 illustrates the major elements comprising the JAF architecture. Note that the framework 60 shown here is not bound to a particular application.

The DataHandler 72 class shown in Figure 4 provides a consistent interface between JAF-aware clients and other subsystems.

The DataSource 70 interface encapsulates an object that contains data, and that can return both a stream providing data access, and a string defining the MIME type describing the data. Classes can be implemented for common data sources (web, file system, IMAP, ftp etc.). The DataSource interface can also be extended to allow per data source user customizations. Once the DataSource is set in the DataHandler, the client can determine the operations available on that data.

The JAF includes two DataSource class implementations for convenience:
FileDataSource accesses data held in a file.
URLDataSource accesses data held at a URL.

The CommandMap 74 provides a service that allows consumers of its interfaces to determine the 'commands' available on a particular MIME Type as well as an interface to retrieve an object that can operate on an object of a particular MIME Type (effectively a component registry). The Command Map can generate and maintain a list of available capabilities on a particular data type by a mechanism defined by the implementation of the particular instance of the CommandMap.

The JavaBeans™ package provides the programming model for the software components that implemented the commands. Each JavaBeans™ component can use externalization, or can implement the CommandObject interface to allow the typed data to be passed to it.

The JAF defines the CommandMap interface, which provides a flexible and extensible framework for the CommandMap. The CommandMap interface allows developers to develop their own solutions for discovering which commands are available on the system. A possible implementation can access the 'types registry' on the platform

or use a server-based solution. The JAF provides a simple default solution based on RFC 1524 (.mailcap) like functionality.

a2

Beans extend the CommandObject 76 interface in order to interact with JAF services. JAF-aware JavaBeans™ components can directly access their DataSource and DataHandler objects in order to retrieve the data type and to act on the data.

Please amend the paragraph from page 14, line 23 to line 28, as shown below.

c3

In these cases, the application can instantiate a DataHandler, using the DataHandler(Object obj, String mimeType) constructor. DataHandler implements the Transferable interface, so the consuming Bean can request representations other than InputStreams. The DataHandler also constructs a DataSource for consumers that request it. The DataContentHandler 78 mechanism is extended to also allow conversion from Objects to InputStreams.